YAHOO! search

Your Search: system/390   [Search]   Advanced Web Search
Preferences

[Web] Directory News NEW! Yellow Pages Images

powered (hp)

SPONSOR RESULTS (What's this?) (Become a Sponsor)

- **System 390 - Compare Price at Nextag.com** - At NexTag.com, the search engine for shoppers, compare prices, tax, shipping, ratings and reviews from name brand stores all over the Web. Sellers, list your own store at Nextag today.
www.nextag.com
- **Access Your Legacy Data with Xbridge** - Access your OS/390 and z/OS legacy data through Windows and Web applications - quickly and easily - with Host Data Connect. Learn how with this free white paper today.
www.xbridgesystems.com

TOP 20 WEB RESULTS out of about 12,100

1. **IBM S/390**
   www.s390.ibm.com/ more results from this site
   ↳ More sites about: Operating Systems > S/390

2. **Eric Loriaux's System/390... MVS, VM, VSE, OS/390, z/OS, Linux/ ...**
   Mainframers meeting point Try the search form I Loriaux's System/390 home page (OS/390, MVS, VSE, VM). You should update your bookmark ...
   www.loriaux.com/s390/ cached | more results from this site

3. **Eric Loriaux's System/390 home page : MVS, VM, VSE, OS/390**
   directory of System/390 resources.
   www.ping.be/~ping1475/ cached | more results from this site
   ↳ More sites about: Systems > IBM S/390

4. **Amazon.com: Books: System 390 Job Control Language, 4th Edition**
   ... 2001 Jetta by Volkswagen. Have one to sell? Don't have one? We'll set one up for you. System 390 Job Control Language, 4th Edition by Gary DeWard Brown (Author). ...
   www.amazon.com/exec/obidos/tg/detail/-/0471283096?v=glance cached | more results from this site

5. **Products > Servers > System/390 Compatible — Fujitsu Technology ...**
   ... Services & Products > Products > Servers > System/390 Compatible. ...
   www.ftsi.fujitsu.com/services/products/compat/ cached | more results from this site

6. **Search390.com**
   resource directory with targeted searches, editorial insight, summaries, daily industry news, and weekly technology tips delivered via email.
   www.search390.com/ cached
   ↳ More sites about: Systems > IBM S/390

7. **System/390 - Wikipedia**
   Not logged in Log in | Help. Other languages: Deutsch. System/390. From

Wikipedia, the free encyclopedia. System/390 is a series of IBM mainframe models.
www.wikipedia.org/wiki/System/390 cached | more results from this site

8. **Mainframe / System/390 / OS/390 - a CompInfo Directory**
   ... Mainframe / System/390 / OS/390 - Outline. ... Mainframe / System/390 / OS/390 - Knowledge Bases. CICS and MQSeries Programmer's Homepage ...
   www.compinfo-center.com/opsys/mainframe.htm cached

9. **System 390**
   ... Support by IBM's unmatched enterprise expertise and services. Several dozen processor models vividly demonstrate the System/390 architecture's dramatic ...
   www.p390.com/s390.html cached | more results from this site

10. **IBM Redbooks | Connectivity on a PC Server System/390**
    ... Contact us. Mailing list. Related links: IBM Publications. Technical Training. Developers. IBM Business Partners. Connectivity on a PC Server System/390 Abstract. . ...
    www.redbooks.ibm.com/abstracts/sg244624.html cached | more results from this site

11. **Return of the Dinosaurs - The IBM System/390 and the Age of ...**
    You Are Here: Guidance » Archives » Return of the Dinosaurs. Return of the Dinosaurs: The IBM System/390 and the Age of Network Computing. ...
    www.guidancecom.com/archives/dino.htm cached

12. **IS Library**
    ... System/390. A potpourri of programming, info, 16 pages, Download this item for free. ... System/390 plans - IBEX survey, info, 31 pages, Download this item for free.
    www.xephon.com/islibrary/islibrary.php?subject=4 cached | more results from this site

13. **THE FUTURE OF SYSTEM/390: SUCCESSES THREATS and REMEDIES (PDF)**
    THE FUTURE OF SYSTEM/390: SUCCESSES THREATS and REMEDIES DCTA, Inc. ... QUERIES AND COMMENTS The Future of System/390 was written and published by DCTA, Inc. ...
    www-3.ibm.com/software/info/university/products/pdfs/s390future.pdf view as html | more results from this site

14. **IBM Archives: System/390 Announcement**
    In its most comprehensive announcement of products, features and functions in more than a quarter century, IBM today introduced System/390 – the industry's ...
    www.ibm.com/ibm/history/exhibits/mainframe/mainframe_PR390.html cached | more results from this site

15. **The end of the low-end IBM System/390 mainframe**
    isham research. The end of the IBM System/390 - survived by zSeries and the Multiprise. 2003 will see the biggest reduction ever in ...
    www.isham-research.com/low_end_s390_2003.html cached | more results from this site

16. **Компьютеры с архитектурой IBM System/390 ...**
    История. Производительность. IBM zSeries. IBM ES/9672. IBM Multiprise 2000. IBM Multiprise 3000. IBM P/390. Старые модели. ...
    www.s390.ru/ cached | more results from this site

17. **Class: 07 DISEASES OF THE CIRCULATORY SYSTEM (390-459)**
    Class: 07 DISEASES OF THE CIRCULATORY SYSTEM (390-459). Below is a list of Categories associated with this Class. To see final diagnoses: ...
    www.eicd.com/EClass/7.htm cached

18. **Results found for: System/390**
    Results found for: System/390. System/390. Search TechWeb For: System/390. The mainframe product line introduced in 1990 by IBM (superseding ...
    www.techweb.com/encyclopedia/defineterm.yb?term=SYSTEM%2F390&exact=1 cached | more results from this site

19. **Kaffe on Linux for System/390**
    Kaffe on Linux for System/390. My former employer has graciously allowed me to post the patches I created to make the Kaffe Java ...
    www.geocities.com/rosspatterson/kaffe/ cached

20. **System 390 (S/390) Mainframe Platform**
    ... System 390 (S/390) Mainframe Platform. ... For more information on System 390 processing rates, visit the DIS Rate Card for details. ...
    techmall.dis.wa.gov/services/sys390.asp cached | more results from this site

MORE SPONSOR RESULTS (What's this?) (Become a Sponsor)

- **OS/390 System Programming Tools** - Powerful, affordable 4GL report writer for OS/390 mainframes. Download your free 30-day trial here.
www.pacsys.com
- **OpenITx: IBM-OS390-L Mail List** - The IBM-OS390-L mail list is for the technical and functional discussion of the OS 390 operating system.
www.openitx.com

Results Page:
1 2 3 4 5 6 7 8 9 10 ▶ Next

[Web] Directory News NEW! Yellow Pages Images

Your Search: system/390   [Search]   Advanced Web Search
Preferences

Search from anywhere online with the Yahoo! Search Toolbar

[Y!] [   ] [Search Web ▾] 🔍 Search This Site ✎ Highlight 📖 Bookmarks

YAHOO! search

Your Search: system/390 "savearea" | **Search** |

Advanced Web Search
Preferences

| **Web** | Directory | News NEW | Yellow Pages | Images |

(i) Did you mean: **system/390 "save area"**?

TOP 7 WEB RESULTS out of about 7

1. **64-Bit z/OS Assembler Coding** (PDF)
   ... IBM, HLASM, OS/390, z/OS, z/Architecture, zSeries and System/390 are trademarks or ... AMODE 31 BASR 12,0 Set a base register USING *,12 LA 13,SAVEAREA Get address ...
   www.tachyonsoft.com/s8158.pdf view as html

2. **2003 Newsgroup postings (3/31 - 4/8) Lynn Wheeler**
   ... HONE was .. Hercules and System/390 - do we need it? http://www ... call rates. they could effectively get by with a static savearea. So I ...
   www.garlic.com/~lynn/2003f.html cached

3. **Condition Handling with Language Environment** (PDF)
   ... ESA MVS MVS/ESA OS/390 VSE/ESA IBM CICS CICS/ESA SYSTEM/390 The information ... Assembler it is no longer appropriate to have the traditional 18F savearea, and you ...
   www.ibm.com/servers/eserver/zseries/zos/le/conference/pdf/smn8234.pdf view as html | more results from this site

4. **High Level Assembler: Toolkit Feature Technical Overview** (PDF)
   ... VM/ESA VSE/ESA OS/2 Warp z/OS z/VM z/Architecture System/390 The following ... r15 Establish Base Using trace,r12 and tell the assembler st r13,savearea+4 Chain la ...
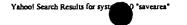   www.share.org/proceedings/sh98/data/S8166A.PDF view as html

5. **CICS Transaction Server for z/OS BigCommareas; How to bypass the ...** (PDF)
   ... Resource Measurement Facility RETAIN RISC System/6000 RMF RT S/370 S/390 SAA SQL/DS SP System/36 System/38 System/360 System/370 System/390 SystemView Systems ...
   www-3.ibm.com/software/htp/cics/library/whitepapers/BigCommarea.pdf view as html

6. **VSE Sept title 2** (PDF)
   ... ALC LA 13,SAVEAREA (13 CAN ALSO BE R13 OR RD). ... BALR 2,Ø USING *,2 ST 13,SAVEAREA+4 LA 13,SAVEAREA LM 3,6,Ø(1) GET ADDRESSES OF PARAMETERS. ...
   www.bimoyle.com/biminfo/pdfmags/sept_98.pdf view as html | more results from this site

7. **CICSPlex SM Messages and Codes** (PDF)
   Page 1. CICS Transaction Server for OS/390 IBM CICSPlex SM Messages and Codes Release 3 GC33-0790-04 Page 2. Page 3. CICS Transaction ...
   examples.oreilly.de/english_examples/cics/CDROM/pdfs/cicsts/eyua100.pdf view as html

---

(i) Did you mean to search for: **system/390 "save area"**?

| **Web** | Directory | News NEW | Yellow Pages | Images |

Your Search: system/390 "savearea" | **Search** |

Advanced Web Search
Preferences

Save time with the Yahoo! Companion Toolbar

Y! [   ] | Search Web ▾ | Search This Site | Highlight | Bookmarks

Programming language considerations

MQSeries provides support for the following programming languages:

- C.
- C++ (MQSeries for AIX, AS/400, HP-UX, OS/2, OS/390, Sun Solaris, Compaq Tru64 UNIX, and Windows NT only). See the *MQSeries Using C++* book for information about coding MQSeries programs in C++.
- Visual Basic (MQSeries for Windows and Windows NT only). See the *MQSeries for Windows Version 2.0 User's Guide* and the *MQSeries for Windows Version 2.1 User's Guide* for information about coding MQSeries programs in Visual Basic.
- COBOL.
- Assembler language (MQSeries for OS/390 only).
- RPG (MQSeries for AS/400 only).
- PL/I (MQSeries for OS/390, AIX, OS/2 Warp, VSE/ESA, and Windows NT only).
- TAL (MQSeries for Tandem NonStop Kernel only).

The call interface, and how you can code the calls in each of these languages, is described in the *MQSeries Application Programming Reference* manual.

MQSeries provides data definition files to assist you with the writing of your applications. For a full description, see Appendix F. MQSeries data definition files.

If you can choose which language to code your programs in, you should consider the maximum length of the messages that your programs will process. If your programs will process only messages of a known maximum length, you can code them in any of the supported programming languages. But if you do not know the maximum length of the messages the programs will have to process, the language you choose will depend on whether you are writing a CICS, IMS, or batch application:

IMS and batch
Code the programs in C, PL/I, or assembler language to use the facilities these languages offer for obtaining and releasing arbitrary amounts of memory. Alternatively, you could code your programs in COBOL, but use assembler language, PL/I, or C subroutines to get and release storage.

CICS
Code the programs in any language supported by CICS. The EXEC CICS interface provides the calls for managing memory, if necessary.

Coding in C

See Appendix A, Language compilers and assemblers for the compilers that you can use to process your C programs.

Note the information in the following sections when coding MQSeries programs in C.

Parameters of the MQI calls

Parameters that are *input-only* and of type MQHCONN, MQHOBJ, or MQLONG are passed by value; for all other parameters, the *address* of the parameter is passed by value.

Not all parameters that are passed by address need to be specified every time a function is invoked. Where a particular parameter is not required, a null pointer can be specified as the parameter on the function invocation, in place of the address of the parameter data. Parameters for which this is possible are identified in the call descriptions.

No parameter is returned as the value of the function; in C terminology, this means that all functions return void.

The attributes of the function are defined by the MQENTRY macro variable; the value of this macro variable depends on the environment.

Parameters with undefined data type

The MQGET, MQPUT, and MQPUT1 functions each have one parameter that has an undefined data type, namely the *Buffer* parameter. This parameter is used to send and receive the application's message data.

Parameters of this sort are shown in the C examples as arrays of MQBYTE. It is valid to declare the parameters in this way, but it is usually more convenient to declare them as the particular structure that describes the layout of the data in the message. The function parameter is declared as a pointer-to-void, and so the address of any sort of data can be specified as the parameter on the function invocation.

Data types

All data types are defined by means of the typedef statement. For each data type, the corresponding pointer data type is also defined. The name of the pointer data type is the name of the elementary or structure data type prefixed with the letter "P" to denote a pointer. The attributes of the pointer are defined by the MQPOINTER macro variable; the value of this macro variable depends on the environment. The following illustrates how pointer data types are declared:

```
#define MQPOINTER                /* depends on environment */
...
typedef MQLONG  MQPOINTER PMQLONG;  /* pointer to MQLONG */
typedef MQMD    MQPOINTER PMQMD;    /* pointer to MQMD */
```

Manipulating binary strings

Strings of binary data are declared as one of the MQBYTEn data types. Whenever you copy, compare, or set fields of this type, use the C functions memcpy, memcmp, or memset:

```
#include <string.h>
#include "cmqc.h"

MQMD MyMsgDesc;

memcpy(MyMsgDesc.MsgId,           /* set "MsgId" field to nulls  */
       MQMI_NONE,                 /* ...using named constant     */
       sizeof(MyMsgDesc.MsgId));
```

```
memset(MyMsgDesc.CorrelId,        /* set "CorrelId" field to nulls */
       0x00,                      /* ...using a different method    */
       sizeof(MQBYTE24));
```

Do not use the string functions strcpy, strcmp, strncpy, or strncmp because these do not work correctly with data declared as MQBYTE24.

Manipulating character strings

When the queue manager returns character data to the application, the queue manager always pads the character data with blanks to the defined length of the field. The queue manager *does not* return null-terminated strings, but you can use them in your input. Therefore, when copying, comparing, or concatenating such strings, use the string functions strncpy, strncmp, or strncat.

Do not use the string functions that require the string to be terminated by a null (strcpy, strcmp, and strcat). Also, do not use the function strlen to determine the length of the string; use instead the sizeof function to determine the length of the field.

Initial values for structures

The include file <cmqc.h> defines various macro variables that may be used to provide initial values for the structures when instances of those structures are declared. These macro variables have names of the form MQxxx_DEFAULT, where MQxxx represents the name of the structure. Use them like this:

```
MQMD   MyMsgDesc = {MQMD_DEFAULT};
MQPMO  MyPutOpts = {MQPMO_DEFAULT};
```

For some character fields, the MQI defines particular values that are valid (for example, for the *StrucId* fields or for the *Format* field in MQMD). For each of the valid values, two macro variables are provided:

- One macro variable defines the value as a string whose length, excluding the implied null, matches exactly the defined length of the field. For example, (the symbol represents a blank character):

```
#define MQMD_STRUC_ID "MD "
#define MQFMT_STRING "MQSTR   "
```

Use this form with the memcpy and memcmp functions.

- The other macro variable defines the value as an array of char; the name of this macro variable is the name of the string form suffixed with "_ARRAY". For example:

```
#define MQMD_STRUC_ID_ARRAY 'M','D',' ',' '
#define MQFMT_STRING_ARRAY 'M','Q','S','T','R',' ',' ',' '
```

Use this form to initialize the field when an instance of the structure is declared with values different from those provided by the MQMD_DEFAULT macro variable.

Initial values for dynamic structures

When a variable number of instances of a structure are required, the instances are usually created in main storage obtained dynamically using the calloc or malloc functions.

To initialize the fields in such structures, the following technique is recommended:

1. Declare an instance of the structure using the appropriate MQxxx_DEFAULT macro variable to initialize the structure. This instance becomes the "model" for other instances:

```
MQMD ModelMsgDesc = {MQMD_DEFAULT};
                                  /* declare model instance */
```

The static or auto keywords can be coded on the declaration in order to give the model instance static or dynamic lifetime, as required.

2. Use the calloc or malloc functions to obtain storage for a dynamic instance of the structure:

```
PMQMD InstancePtr;
InstancePtr = malloc(sizeof(MQMD));
                                  /* get storage for dynamic instance */
```

3. Use the memcpy function to copy the model instance to the dynamic instance:

```
memcpy(InstancePtr, &ModelMsgDesc, sizeof(MQMD));
                                  /* initialize dynamic instance */
```

Use from C++

For the C++ programming language, the header files contain the following additional statements that are included only when a C++ compiler is used:

```
#ifdef __cplusplus
  extern "C" {
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

Coding in COBOL

See Appendix A, Language compilers and assemblers for the compilers that you can use to process your COBOL programs.

Note the information in the following sections when coding MQSeries programs in COBOL.

Named constants

In this book, the names of constants are shown containing the underscore character (_) as part of the name. In COBOL, you must use the hyphen character (-) in place of the underscore.

Constants that have character-string values use the single quotation mark character (') as the string delimiter. To make the compiler accept this character, use the compiler option APOST.

The copy file CMQV contains declarations of the named constants as level-10 items. To use the constants, declare the level-01 item explicitly, then use the COPY statement to copy in the declarations of the constants:

```
WORKING-STORAGE SECTION.
01  MQM-CONSTANTS.
    COPY CMQV.
```

However, this method causes the constants to occupy storage in the program even if they are not referred to. If the constants are included in many separate programs within the same run unit, multiple copies of the constants will exist--this may result in a significant amount of main storage being used. You can avoid this situation by adding the GLOBAL clause to the level-01 declaration:

```
* Declare a global structure to hold the constants
01  MQM-CONSTANTS GLOBAL.
    COPY CMQV.
```

This causes storage to be allocated for only *one* set of constants within the run unit; the constants, however, can be referred to by *any* program within the run unit, not just the program that contains the level-01 declaration.

Coding in System/390(R) assembler language

System/390 assembler is supported on OS/390 only.

See Appendix A, Language compilers and assemblers for the assemblers that you can use to process your assembler-language programs.

Note the information in the following sections when coding MQSeries for OS/390 programs in assembler language.

Names

In this book, the names of parameters in the descriptions of calls, and the names of fields in the descriptions of structures are shown in mixed case. In the assembler-language macros supplied with MQSeries, all names are in uppercase.

Using the MQI calls

The MQI is a call interface, so assembler-language programs must observe the OS linkage convention. In particular, before they issue an MQI call, assembler-language programs must point register R13 at a save area of at least 18 full words. This save area is to provide storage for the called program. It stores the registers of the caller before their contents are destroyed, and restores the contents of the caller's registers on return.
Note: This is of particular importance for CICS assembler-language programs that use the DFHEIENT macro to set up their dynamic storage, but that choose to override the default DATAREG from R13 to other registers. When the CICS Resource Manager Interface receives control from the

stub, it saves the current contents of the registers at the address to which R13 is pointing. Failing to reserve a proper save area for this purpose gives unpredictable results, and will probably cause an abend in CICS.

Declaring constants

Most constants are declared as equates in macro CMQA. However, the following constants cannot be defined as equates, and these are not included when you call the macro using default options:

```
    MQACT_NONE
    MQCI_NONE
    MQFMT_NONE
    MQFMT_ADMIN
    MQFMT_COMMAND_1
    MQFMT_COMMAND_2
    MQFMT_DEAD_LETTER_HEADER
    MQFMT_EVENT
    MQFMT_IMS
    MQFMT_IMS_VAR_STRING
    MQFMT_PCF
    MQFMT_STRING
    MQFMT_TRIGGER
    MQFMT_XMIT_Q_HEADER
    MQMI_NONE
```

To include them, add the keyword EQUONLY=NO when you call the macro.

CMQA is protected against multiple declaration, so you can include it many times. However, the keyword EQUONLY takes effect only the first time the macro is included.

Specifying the name of a structure

To allow more than one instance of a structure to be declared, the macro that generates the structure prefixes the name of each field with a user-specifiable string and an underscore character (_). Specify the string when you invoke the macro. If you do not specify a string, the macro uses the name of the structure to construct the prefix:

```
* Declare two object descriptors
        CMQODA              Prefix used="MQOD_" (the default)
MY_MQOD  CMQODA              Prefix used="MY_MQOD_"
```

The structure declarations in the *MQSeries Application Programming Reference* manual show the default prefix.

Specifying the form of a structure

The macros can generate structure declarations in one of two forms, controlled by the DSECT parameter:

DSECT=YES	An assembler-language DSECT instruction is used to start a new data section; the structure definition immediately follows the DSECT statement. No storage is allocated, so no initialization is possible. The label on the macro invocation is used as the name of the data section; if no label is specified, the name of the structure is used.
DSECT=NO	Assembler-language DC instructions are used to define the structure at the current position in the routine. The fields are initialized with values, which you can specify by coding the relevant parameters on the macro invocation. Fields for which no values are specified on the macro invocation are initialized with default values.

DSECT=NO is assumed if the DSECT parameter is not specified.

Controlling the listing

You can control the appearance of the structure declaration in the assembler-language listing by means of the LIST parameter:

LIST=YES	The structure declaration appears in the assembler-language listing.
LIST=NO	The structure declaration does not appear in the assembler-language listing. This is assumed if the LIST parameter is not specified.

Specifying initial values for fields

You can specify the value to be used to initialize a field in a structure by coding the name of that field (without the prefix) as a parameter on the macro invocation, accompanied by the value required.

For example, to declare a message descriptor structure with the *MsgType* field initialized with MQMT_REQUEST, and the *ReplyToQ* field initialized with the string MY_REPLY_TO_QUEUE, you could use the following code:

```
MY_MQMD     CMQMDA          MSGTYPE=MQMT_REQUEST,            X
            REPLYTOQ=MY_REPLY_TO_QUEUE
```

If you specify a named constant (or equate) as a value on the macro invocation, you must use the CMQA macro to define the named constant. You must not enclose in single quotation marks (' ') values that are character strings.

Writing reenterable programs

MQSeries uses its structures for both input and output. If you want your program to remain reenterable, you should:

1. Define working storage versions of the structures as DSECTs, or define the structures inline within an already-defined DSECT. Then copy the DSECT to storage that is obtained using:
   o For batch and TSO programs, the STORAGE or GETMAIN OS/390 assembler macros
   o For CICS, the working storage DSECT (DFHEISTG) or the EXEC CICS GETMAIN command

To correctly initialize these working storage structures, copy a constant version of the corresponding structure to the working storage version.
Note: The MQMD and MQXQH structures are each more than 256 bytes long. To copy these structures to storage, you will have to use the MVCL assembler instruction.

2. Reserve space in storage by using the LIST form (MF=L) of the CALL macro. When you use the CALL macro to make an MQI call, use the EXECUTE form (MF=E) of the macro, using the storage reserved earlier, as shown in the example under "Using CEDF". For more examples of how to do this, see the assembler language sample programs as shipped with MQSeries.

Use the assembler language RENT option to help you determine if your program is reenterable.

For information on writing reenterable programs, see the *MVS/ESA Application Development Guide: Assembler Language Programs*, GC28-1644.

Using CEDF

If you want to use the CICS-supplied transaction, CEDF (CICS Execution Diagnostic Facility) to help you to debug your program, you must add the , VL keyword to each CALL statement, for example:

```
CALL MQCONN, (NAME, HCONN, COMPCODE, REASON), MF=(E, PARMAREA), VL
```

The above example is reenterable assembler-language code where PARMAREA is an area in the working storage you specified.

Coding in RPG

RPG is supported on OS/400 only.

See Appendix A, Language compilers and assemblers for the compilers that you can use to process your RPG programs.

In this book, the parameters of calls, the names of data types, the fields of structures, and the names of constants are described using their long names. In RPG, these names are abbreviated to six or fewer uppercase characters. For example, the field *MsgType* becomes *MDMT* in RPG. For more information, see the *MQSeries for AS/400 Application Programming Reference (ILE RPG)* manual.

Coding in PL/I

PL/I is supported on AIX, OS/390, OS/2 Warp, VSE/ESA, and Windows NT only.

See Appendix A, Language compilers and assemblers for the compilers that you can use to process your PL/I programs.

Note the information in the following sections when coding MQSeries for OS/390 programs in PL/I.

Structures

Structures are declared with the BASED attribute, and so do not occupy any storage unless the program

declares one or more instances of a structure.

An instance of a structure can be declared by using the like attribute, for example:

```
dcl my_mqmd        like MQMD;  /* one instance */
dcl my_other_mqmd like MQMD;  /* another one */
```

The structure fields are declared with the INITIAL attribute; when the like attribute is used to declare an instance of a structure, that instance inherits the initial values defined for that structure. Thus it is necessary to set only those fields where the value required is different from the initial value.

PL/I is not sensitive to case, and so the names of calls, structure fields, and constants can be coded in lowercase, uppercase, or mixed case.

Named constants

The named constants are declared as macro variables; as a result, named constants which are not referenced by the program do not occupy any storage in the compiled procedure. However, the compiler option which causes the source to be processed by the macro preprocessor must be specified when the program is compiled.

All of the macro variables are character variables, even the ones which represent numeric values. Although this may seem counter intuitive, it does not result in any data-type conflict after the macro variables have been substituted by the macro processor, for example:

```
%dcl MQMD_STRUC_ID char;
%MQMD_STRUC_ID = '''MD ''';

%dcl MQMD_VERSION_1 char;
%MQMD_VERSION_1 = '1';
```

Coding in TAL

TAL is supported on Tandem NonStop Kernel only.

See Appendix A. Language compilers and assemblers for the compilers that you can use to process your TAL programs.

Note the following when coding MQSeries for Tandem NonStop Kernel programs in TAL:

- The MQI library (bound into the application process) does not open $RECEIVE and does not open $TMP (TM/MP transaction pseudo-file) itself, so you may code your application to use these features.
- The MQI library uses a SERVERCLASS_SEND_() call in initial communication with the Queue Manager. While connected, it maintains two process file opens (with the LINKMON process and a Local Queue Manager Agent) and a small number of disk file opens (fewer than 10).